DURÉE: 1h30, UNE FEUILLE A5 RECTO-VERSO MANUSCRITE AUTORISÉE

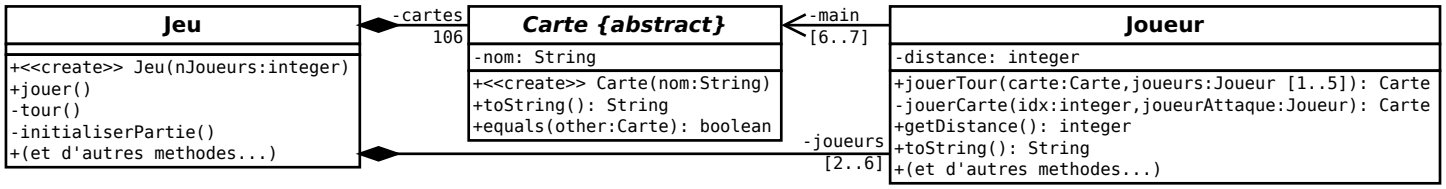
REMARQUES:
- Les exercices sont indépendants et peuvent être réalisés dans l'ordre voulu.
- Dans l'implémentation d'une méthode, on pourra utiliser n'importe quelle autre méthode définie auparavant même si celle-ci n'a pas été implémentée.
- Dans toutes les implémentations que vous écrivez, pensez à respecter le guide de syntaxe pour la programmation (règles de nommage, présentation des blocs, *etc.*).

EXERCICE 1: ALGORITHMIE (6 POINTS)

1. Écrire une fonction `combienDeMots( chaine )` qui compte le nombre de mots dans une chaîne de caractères.
   - On supposera que les mots sont séparés par des espaces.
   - La chaîne pourra contenir les caractères de ponctuation suivant : `,.!?;:'"-` et on s'assurera donc qu'un mot contient toujours au moins une lettre.
     Par exemple, la phrase `Einstein disait : " Deux choses sont infinies : l'Univers et la bêtise humaine. "` contient 11 mots.
   - On rappelle qu'il existe une fonction `isalpha` qui vérifie si tous les caractères d'une chaine sont des lettres.
2. Écrire une fonction `isStrOk( phrase )` qui renvoie un booléen indiquant si une chaîne de caractères ne contient que des lettres ou des caractères de ponctuation. La phrase ne devra pas être modifiée.
3. Écrire une fonction `str2Tab( phrase )` qui renvoie un tableau contenant chaque lettre d'une chaîne de caractères. Les lettres devront toutes être en minuscule et les caractères de ponctuation ne devront pas apparaître dans le tableau. La phrase ne devra pas être modifiée.
4. Écrire une fonction `isAnagramme( phrase1, phrase2 )` qui renvoie un booléen indiquant si les 2 chaînes de caractères passées en entrée sont des anagrammes.
   Un anagramme est un mot ou une phrase constitué des mêmes lettres qu'un autre mot ou phrase. Par exemple *Carpe Diem* et *Ça déprime* ou *Tom Marvolo Riddle* et *I am Lord Voldemort* sont des anagrammes. Si l'une des phrases n'est pas uniquement constituée de lettres ou de caractères de ponctuation, on renverra faux. On supposera aussi qu'aucun caractère accentué n'est présent.
   On rappelle qu'il existe en Python une fonction `sorted(tab)` qui renvoie une version triée d'un tableau `tab` (sans le modifier).
5. Écrire un programme principal qui :
   - déclare 2 chaînes de caractères "Le boson scalaire de Higgs" et "L'horloge des anges ici-bas",
   - compte et affiche le nombre de mots de la deuxième chaîne de caractères,
   - vérifie et affiche (de manière claire) si les 2 chaînes de caractères sont des anagrammes

EXERCICE 2: "1000 BORNES™" (6 POINTS)

Le diagramme UML ci-dessous présente une modélisation partielle d'un jeu de "1000 Bornes™" .

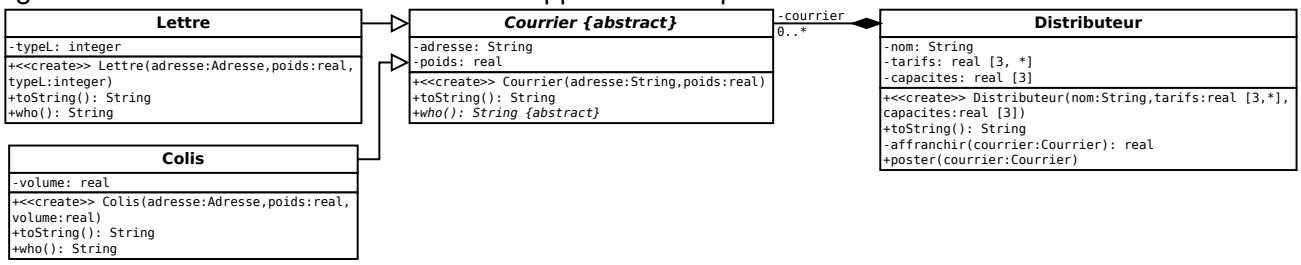


Dans la classe `Jeu`, la méthode `jouer` distribue les Cartes aux Joueurs (créés au moment de la construction du `Jeu`) puis les fait jouer à tour de rôle jusqu'à ce qu'il y ait un vainqueur. La méthode `tour` permet d'effectuer un tour de jeu (*i.e.* de faire jouer une fois chaque joueur).

1. Pourquoi la classe `Carte` est-elle abstraite alors qu'aucune de ses méthodes ne l'est ?
2. Quelle est la relation entre la classe `Jeu` et la classe `Carte` ? Justifier ce choix.
3. Pourquoi la relation entre `Joueur` et `Carte` n'est-elle pas la même que celle entre `Jeu` et `Carte` ?
4. Pourquoi la méthode `jouer` de `Jeu` est-elle publique alors que la méthode `tour` est privée ?
5. Donner l'implémentation de la méthode `+jouer( )`. Cette méthode peut être implémentée en appelant uniquement les méthodes présentes dans le diagramme UML. La méthode affichera l'état des `Joueurs` au début de chaque tour et affichera le vainqueur à la fin de la partie.

EXERCICE 3: PROGRAMMATION ORIENTÉE OBJET (10 POINTS)

Le diagramme UML ci-dessous modélise une application simplifiée d'envoi du courrier.



Un `Courrier` sera représenté par l'adresse du destinataire et son poids. La méthode `who( )` renverra le nom de la classe. L'attribut `typeL` de `Lettre` pourra être de type "normal" (0), "prioritaire" (1) ou avec accusé de réception AR (2). Pour un `Colis`, on conservera son volume en plus de son poids.

Un `Distributeur` sera modélisé par son nom (LaPoste, DHL, ...), les tarifs qu'il applique, les capacités avant envoi de sa flotte et les courriers qu'il doit distribuer. On considèrera que l'envoi d'un courrier passe par 2 étapes : il est d'abord posté (et payé) par le client puis lorsque l'une des capacités du distributeur est dépassée, l'ensemble du courrier reçu par celui-ci est envoyé. Les 3 lignes de l'attribut `tarifs` contiennent respectivement :

1. 2 cases : le prix au kilo pour une lettre ou un colis
2. les prix en fonction du type de la lettre (normal, prioritaire ou avec AR)
3. les prix au m^3 pour des colis de moins de 0.01m^3, de moins de 0.25m^3 ou de plus de 0.25m^3.

L'attribut `capacites` est un tableau de 3 cases contenant le nombre (case 1), le poids (case 2) et le volume (case 3) maximaux de courriers. Dès que l'un de ces 3 critères n'est plus respecté, l'ensemble des courriers reçus est distribué.

Un exemple d'affichage (correspondant au programme principal de la question 3) est donné ci-dessous et on veillera à respecter ces consignes.

```
Lettre destine(e) a Chuck Norris, 5 place Jussieu, 75005 Paris // Poids: 0.17kg, Type: normal
Colis destine(e) a Darth Vador, 1ere a droite, Une galaxie fort fort lointaine // Poids: 7.55kg,
    Volume: 1.70m^3
Lettre destine(e) a Hairy Otter, Quai 9 3/4, 87160 Arnac-la-Poste // Poids: 0.30kg, Type: AR
Courrier poste. Prix: 122.63 euros
Courrier poste. Prix: 5.59 euros
Courrier poste. Prix: 0.34 euros
DHL: 2 lettres et 1 colis pour un poids de 8.01kg et un volume de 1.70m^3.
```

1. **Classes** `Courrier`**,** `Lettre` **et** `Colis` :
   (a) Implémenter la classe `Courrier` (import, attributs, méthodes).
   (b) Pourquoi la méthode `who` est-elle abstraite ? Quelle(s) conséquence(s), cela a t'il sur la classe `Courrier` ?
   (c) Implémenter la classe `Lettre`.
2. **Classe** `Distributeur` :
   (a) Implémenter la déclaration de la classe `Distributeur` ainsi que le constructeur et la méthode `toString`. On affichera le nom, le nombre de lettres, le nombre de colis, le poids total des courriers à envoyer ainsi que le volume des colis.
   (b) Implémenter la fonction `-affranchir( courrier: Courrier ): real`, qui renvoie le prix pour envoyer le `courrier`. Le prix sera calculé de la manière suivante :
   — pour une `Lettre` : (prix au kilo d'une lettre * poids de la lettre) + prix en fonction du type de la lettre
   — pour un `Colis` : (prix au kilo d'un colis * poids du colis) + (prix au m^3 * volume du colis)
   (c) Implémenter la fonction `+poster( courrier: Courrier )`, qui
   — affiche un message indiquant que le `courrier` est reçu et le prix pour son envoi
   — l'ajoute au courrier déjà reçu
   — envoie l'ensemble du courrier si l'une des `capacites` est dépassée
3. **Programme principal** : écrire un programme principal qui génère des `Courriers` et les ajoute dans un tableau, affiche l'ensemble des `Courriers`, les poste auprès d'un `Distributeur` puis affiche le `Distributeur`. Pour la génération des Courriers
   — on effectuera un tirage aléatoire pour déterminer le type (`Lettre` ou `Colis`) de `Courrier`,
   — une fonction `genererAdresse( ) : String` existe et permettra d'obtenir l'adresse du destinataire,
   — on génèrera aléatoirement le poids (entre 0.1 et 0.5kg) et le type (normal, ...) d'une `Lettre`,
   — on génèrera aléatoirement le poids (entre 0.5 et 10kg) et le volume (entre 0.005 et 2m^3) d'un `Colis`